**IN THE CLAIMS:**

Please amend the claims as follows:

1. (Currently Amended) A graphics processor configured to produce data for multiple output buffers, each output buffer associated with a unique output buffer identifier, comprising:

a fragment processing pipeline configured to process graphics data to produce processed graphics data for the multiple output buffers and determine at least one output buffer identifier associated with the processed graphics data, the fragment processing pipeline determining an address associated with the processed fragment data, the address corresponding to a specific location within an output buffer;

a shader read interface configured to read processed graphics data associated with an output buffer identifier from an output buffer stored in a memory; and

a write interface configured to write processed graphics data associated with at least one output buffer identifier to an output buffer stored in the memory.

2. (Original) The graphics processor of claim 1, further comprising a geometry processor configured to process graphics data.

3. (Cancelled)

4. (Currently Amended) The graphics processor of claim 1, wherein an output buffer includes data represented in two or more data formats including fixed point and floating point or including different numbers of bits within the same output buffer.

5. (Original) The graphics processor of claim 1, wherein the output buffer identifier is readable and writable by the fragment processing pipeline.

6. (Currently Amended) The graphics processor of claim 1, wherein any of the multiple output buffers is selected for display using the address defined in the fragment processing pipeline.

7. (Original) The graphics processor of claim 1, wherein the fragment processing pipeline includes multiple registers, each register capable of outputting data to one or more of the multiple output buffers.

8. (Cancelled)

9. (Original) The graphics processor of claim 1, wherein the graphics processor resides within a computing system including a host processor.

10. (Original) A method of processing fragment data for multiple output buffers, comprising:

processing fragment data as specified by a fragment program to produce processed fragment data for the multiple output buffers;

determining an output buffer identifier associated with the processed fragment data; and

storing the processed fragment data in an output buffer corresponding to the output buffer identifier.

11. (Original) The method of claim 10, further comprising:

reading the processed fragment data from the output buffer using the output buffer identifier.

12. (Original) The method of claim 10, wherein the processed fragment data stored in the output buffer includes fragment depth data.

13.-15. (Cancelled)

16. (Currently Amended) The method of claim 10, wherein the processed fragment data stored in the output buffer includes displaced mesh data, the method including displacing the processed fragment data to produce one or more displaced meshes, and storing each of the displaced meshes in one of the multiple output buffers.

17.-20. (Cancelled)

21. (New) The graphics processor of claim 1, wherein the output buffer includes data having different multi-bit formats including 16-bit and 32-bit formats within the same output buffer.

22. (New) The graphics processor of claim 1, wherein the graphics data includes fragment data including at least two fragments within a single surface, the fragment processing pipeline being configured to separate the at least two fragments and write each of the fragments to a separate one of the output buffers.

23. (New) The graphics processor of claim 22, wherein the fragment processing pipeline is configured to select one of the output buffers for writing each of the fragments based on a procedurally computed function.

24. (New) The graphics processor of claim 1, wherein the fragment processing pipeline is configured to process the graphics data of at least two fragments in parallel, the fragment processing unit determining the destination address in one of the output buffers for each of the fragments.

25. (New) The graphics processor of claim 24, wherein the fragment processing pipeline is configured to respond to a flush instruction to avoid read after write conflicts when reading from one of the output buffers that can be accessed by the parallel processing, the processor responding to a write instruction to indicate that all pending write operations are complete.

26. (New) The graphics processor of claim 1, wherein the fragment processing pipeline is configured to process graphics data to produce visible fragment data for visible fragments;

one of multiple output buffers stores depth map values of the visible fragments;

one or more of the output buffers store fragment data; and

the graphics processor further includes a shader for shading the visible fragments using the portion of the fragment data read from the one or more of the output buffers and the depth map read from one of the output buffers.

27. (New) The method of claim 12, wherein the processed fragment data includes visible fragment data for visible fragments;

one of multiple output buffers stores the depth map data of the visible fragments; and

shading the visible fragments using the portion of the fragment data read from the one or more additional output buffers and the depth map data read from one of the output buffers.

28. (New) The method of claim 27, wherein each of the displaced meshes is used as a vertex array to animate geometry.

29. (New) The method of claim 28, further including displacing the processed fragment data along a normal vector to produce the displaced meshes.